How to Build Your First OpenAI Agent

Video: https://youtu.be/bG7pHH4rBHM

Page 1: Welcome! Your First AI Agent Awaits!

A. What is an AI Agent? (And Why It's Exciting!)

Imagine having a smart assistant that can understand instructions, perform tasks, and even learn. That's the essence of an Artificial Intelligence (AI) Agent. Think of it not just as a piece of software, but as a digital helper designed to achieve specific goals. These agents can range from simple bots that answer questions to complex systems that manage intricate workflows.

The excitement around AI agents comes from their potential to interact with the digital world in intelligent ways. Building one, even a simple one, is like teaching a computer to "think" and act on a user's behalf. In this guide, the goal is to build a first AI agent that can fetch real-time information – specifically, to find out the weather. This hands-on experience will demystify the process and show that creating a basic AI agent is within reach.

B. What Users Will Accomplish Today

This tutorial is designed for beginners, so no prior coding expertise is assumed. By following the steps, users will:

- Set up a free, online coding environment called Google Colab.
- Install the necessary software library for building OpenAI agents.
- Obtain and securely use a special access code called an OpenAI API key.
- Write a few lines of simple code to define and instruct an AI agent.
- Run the agent and see it perform its task: answering a weather question.

The journey is broken down into small, manageable steps, making the process straightforward and rewarding.

C. What Users Will Need (Just a Few Things!)

To embark on this AI agent-building adventure, only a few items are required:

- **A Google Account:** This is needed to access Google Colab, the online platform where the agent will be built.
- **An OpenAI Account:** This is necessary to get an API key, which allows the agent to use OpenAI's powerful AI models.
- **A Dash of Curiosity:** An eagerness to learn and explore is the most important ingredient!

D. The Adventure Map: Our Workflow

To provide a clear overview of the journey, here’s a simple map of the process. This

workflow outlines the main stages involved in building the first AI agent.

Workflow Illustration: Your AI Agent Building Journey!

```
[16] pip install openai-agents
```

```
[17] from agents import Agent, Runner, WebSearchTool
     import os

     from google.colab import userdata


     os.environ["OPENAI_API_KEY"] = userdata.get('DjamgatechAPIKey2OpenAI')
```

```
[18] weather_agent = Agent(
         name="Weather agent",
         instructions="You answer weather questions.",
         model="gpt-4o",
         tools=[WebSearchTool()],
     )
```

```
result = await Runner.run(weather_agent, "What's the weather in Calgary?")
print(result.final_output)
```

The workflow can be visualized as a sequence of six key steps, each leading to the next:

1. **Set Up Your Playground:** This step involves accessing Google Colab. Think of it as preparing the workshop or kitchen where the creation will happen. (Visually, this could be represented by a Google Colab icon or a simple notebook icon).
2. **Install Special Tools:** Here, the openai-agents library is installed. This is like getting the specific toolkit or ingredient set needed for this particular recipe. (An icon representing a software package or a toolbox would be suitable).
3. **Get Your Secret Key:** This refers to obtaining the OpenAI API Key. This key is essential for the agent to access OpenAI's AI capabilities. (A key icon, perhaps with the OpenAI logo, would illustrate this).
4. **Tell Your Playground the Key:** The API key needs to be made available to the

Colab environment in a secure way. This is like unlocking the special features of the workshop. (A code snippet icon or an icon representing a secure lockbox or environment variable would fit here).

5. **Build Your Agent:** This is where the actual code for the agent is written – defining its name, instructions, and tools. This is the core creation phase. (A robot icon or gears turning would symbolize this).
6. **Ask & See the Magic!:** Finally, the agent is run with a specific question, and the results are observed. This is the moment of truth where the creation comes to life. (A speech bubble interacting with a magnifying glass or a "play" button leading to an output display would be appropriate).

This map provides a bird's-eye view, and each step will be detailed in the following pages.

Page 2: Step 1 & 2 – Setting the Stage

A. Step 1: Your Coding Playground – Google Colab

The first step is to set up the environment where the AI agent will be built. For this, Google Colab will be used. Google Colab is a free, online service that allows users to write and execute Python code through their browser. It's essentially a Jupyter notebook hosted in the cloud, meaning no installation is required on the user's computer, making it very convenient.[1]

To get started:

1. Open a web browser and go to the Google Colab website (colab.research.google.com).
2. If prompted, sign in with a Google account.
3. Once on the Colab welcome page, look for an option to create a "New notebook." Clicking this will open a fresh, untitled notebook.

Colab notebooks are made up of "cells." There are code cells, where Python code is typed and run, and text cells, where notes and explanations (like this guide) can be written.[1]

B. Step 1B: Naming Your Project

Once a new notebook is open, it's a good practice to give it a descriptive name. By default, it might be named something like "Untitled0.ipynb".

1. Click on the default file name at the top-left of the Colab interface.
2. A dialog box will appear, allowing the name to be changed.
3. Rename it to something like "MyFirstOpenAIAgent" and press Enter.

Naming projects helps in keeping work organized, especially as one starts to create more notebooks.

C. Step 1C: Installing the "Agent Building Kit"

To build an OpenAI agent, a specific Python library called openai-agents is needed. This library provides the tools and functions to create and manage agents.[2] In Colab, libraries that are not pre-installed can be added using a command called pip install.[1]

1. In the first code cell of the Colab notebook, type the following command:

```
!pip install openai-agents
```

2. To run this cell, click the "play" button to the left of the cell, or press Shift+Enter.
The ! at the beginning of the command tells Colab that this is a shell command, not Python code. This command will download and install the openai-agents library into the Colab environment for the current session.

D. Step 2: Getting Your OpenAI API Key – The "Magic Wand"

An API (Application Programming Interface) key is a unique code that grants access to a service. In this case, an OpenAI API key is needed to allow the agent to use OpenAI's powerful language models, like GPT-4o.[4] It's like a secret password that proves the application is authorized to use the service.

To get an API key, an OpenAI account is required.

1. Visit the OpenAI website (platform.openai.com).
2. If an account already exists, log in. If not, sign up for a new account. The sign-up process might involve providing an email address, creating a password, or using a Google or Microsoft account for convenience.[4] New users may also need to verify their phone number.[5]

E. Step 2B: Navigating to API Keys

Once logged into the OpenAI account on the platform website:

1. Look for a section typically labeled "API Keys" in the dashboard or navigation menu on the left-hand side.[4] It might be under a "Developers" or "Settings" area.

F. Step 2C: Adding Some "Power" (Billing/Credits)

Using OpenAI's API, especially for advanced models, typically involves costs based on usage. OpenAI often provides new users with some free credits to get started.[4] However, to ensure uninterrupted access and to use the API beyond any initial free tier, a payment method needs to be set up.

1. In the OpenAI platform dashboard, find the "Billing" section.[4]
2. Follow the prompts to "Add payment details" and enter valid credit or debit card information.[5]
3. Users can often choose how much credit to purchase or set up automatic recharges.[5] It's important to review OpenAI's pricing details to understand the costs associated with different models and usage levels.

Even if starting with free credits, setting up billing is often a prerequisite for generating an API key.[4]

G. Step 2D: Creating and Copying Your Secret Key

With the account set up and billing information provided (if required):

1. Navigate back to the "API Keys" section.
2. Click on a button that says "Create new secret key" or similar.[4]
3. A dialog box may ask for an optional name for the key (e.g., "MyFirstAgentKey"). This helps in identifying the key's purpose later.
4. Once the key is generated, it will be displayed on the screen. **This is the only time the full key will be shown.**
5. Copy the API key immediately and paste it into a secure, temporary location (like a simple text editor on the computer). Do not share this key publicly or embed it directly in shared code repositories.[4]

Losing the key means a new one will have to be generated, as the old one cannot be retrieved again.[5]

Page 3: Step 3 – Bringing Your Agent to Life in Colab (Part 1)

A. Step 3A: Telling Colab Your Secret Key (Securely!)

Now that the OpenAI API key is obtained, the Colab environment needs to be informed of this key so it can communicate with OpenAI's services. It is a best practice to not paste the key directly into the main code that might be shared. Instead, it can be set

as an "environment variable." This is a way to store sensitive information that the code can access without the key itself being visible in the script.

The os library in Python helps manage environment variables.[3]

1. In a new code cell in the Colab notebook, type the following Python code:
```
Python
import os
os.environ["OPENAI_API_KEY"] = "YOUR_API_KEY_HERE"
```
2. **Crucially, replace "YOUR_API_KEY_HERE" with the actual API key copied from the OpenAI platform.** Ensure the key is enclosed in quotation marks.
3. Run this cell (click the play button or press Shift+Enter).

The first line, import os, makes Python's operating system interface tools available. The second line creates an environment variable named OPENAI_API_KEY and assigns the provided API key string to it. The openai-agents library will automatically look for this environment variable to authenticate requests.[3] This method helps keep the API key more secure than hardcoding it directly where the agent logic is defined.

Page 4: Step 3 – Bringing Your Agent to Life in Colab (Part 2)

A. Step 3B: Importing the Agent's Building Blocks

With the API key set up, the next step is to import the necessary components from the openai-agents library that was installed earlier. Think of this like a chef gathering all the specific ingredients and tools before starting to cook. These imports make the classes and functions needed to define and run the agent available to the code.

1. In a new code cell, type the following Python code:
```
Python
from agents import Agent, Runner, WebSearchTool
# import os # This was done in the previous step to set the API key.
        # It's good practice to group imports, but not strictly necessary to re-import here.
```
2. Run this cell.

Let's briefly understand what is being imported:

- Agent: This is the fundamental building block, representing the AI agent itself. It will be configured with instructions, a model, and tools.[2]
- Runner: This component is used to execute the agent, essentially making it

perform its task with a given input.[2]
- WebSearchTool: This is a pre-built tool that allows the agent to search the web for information. This is what will enable the weather agent to find current weather data.[6]

B. Step 3C: Creating Your "Weather Agent"

Now for the exciting part: defining the agent! This involves creating an instance of the Agent class and telling it what its name is, what its job (instructions) should be, which AI "brain" (model) to use, and what special abilities (tools) it has.

1. In a new code cell, type the following Python code. This code is based on the structure shown in the user-provided image.
```
Python
weather_agent = Agent(
    name="weather_agent",
    instructions="You answer weather questions.",
    model="gpt-4o",
    tools=
)
```

2. Run this cell. This creates the agent object in memory, but it doesn't do anything yet.

Let's break down what each part of this code does:

- name="weather_agent": This is simply a friendly name given to this specific agent instance.
- instructions="You answer weather questions.": This is a critical piece. These instructions define the agent's primary goal and how it should behave. Clear, concise instructions lead to better agent performance. The agent will try to adhere to this directive.
- model="gpt-4o": This specifies which of OpenAI's language models the agent should use as its "brain." gpt-4o is a very capable and recent model that supports tool use, including web search.[7] Other models like gpt-4o-mini or gpt-3.5-turbo could also be used, potentially offering different balances of speed, cost, and capability.[3] For this tutorial, gpt-4o as shown in the example image is a good choice.
- tools=: This equips the agent with the WebSearchTool. By including this, the agent gains the ability to perform web searches to find information needed to answer questions, such as current weather conditions.[6]

C. Step 3D: Asking Your Agent a Question

The weather_agent is now defined and configured. To make it work, the Runner component is used. The Runner takes the agent and an input (the question) and manages the execution.

1. In a new code cell, type the following Python code, which is also derived from the example image:
   ```
   Python
   result = await Runner.run(weather_agent, "What's the weather in NYC?")
   ```
2. Run this cell. This step might take a few moments as the agent processes the request, potentially uses the WebSearchTool to find information, and formulates a response.

A quick note on await: The await keyword is used in Python for asynchronous operations, which means the program can work on other things while waiting for a long task (like an API call or web search) to complete. Google Colab notebooks generally handle top-level await in cells quite well. If an error message like "await outside async function" appears, one common solution is to put the agent definition code (from Step 3C) and this Runner.run code into the *same* Colab cell and run that combined cell.

This line of code tells the Runner to execute the weather_agent with the input question "What's the weather in NYC?". The outcome of this execution will be stored in the variable result.

D. Step 3E: Seeing the Answer!

The agent has now processed the question and (hopefully) found an answer. This answer is contained within the result object. Specifically, the openai-agents library stores the agent's final textual output in an attribute called final_output.[2]

1. In a new code cell, type:
   ```
   Python
   print(result.final_output)
   ```
2. Run this cell.

This command will display the text of the agent's response. If everything worked correctly, users should see the weather information for New York City!

Page 5: Success! Understanding the Results & What's Next

A. What Your Agent Said (The Output Explained)

After running the print(result.final_output) command, the Colab notebook should display the agent's response. This output will likely be similar to the example shown in the black box of the initial tutorial image. It should contain current weather conditions for New York City, possibly including the temperature, a brief description (e.g., "light rain," "sunny"), and perhaps a short forecast.

This information wasn't just pulled out of thin air by the agent. It actively used the WebSearchTool that it was equipped with.[6] When the agent received the question, it understood from its instructions ("You answer weather questions.") and the nature of the query that it needed up-to-date information. It then utilized the WebSearchTool to query the internet, find relevant weather data for NYC, and then formulated the answer presented as final_output. This demonstrates the power of giving agents tools to interact with external sources of information.

B. 🎉 Congratulations! Users Have Built Their First AI Agent! 🎉

A moment to celebrate is in order! By reaching this point, users have successfully navigated several key steps in the world of AI development:

- They have set up and used Google Colab, a powerful online coding environment.
- They have installed a specialized AI software library.
- They have obtained and securely configured an OpenAI API key.
- They have written the Python code to define an AI agent with specific instructions, a model, and a tool.
- And most importantly, they have made their agent perform a task and deliver a result!

This is a significant first step into the practical application of AI and provides a solid foundation for further exploration.

C. Quick Troubleshooting (If Things Went Wrong)

Sometimes, even with careful steps, things might not go as planned. Here are a few common issues and how to address them:

- **Error related to "API Key"?**
  - Symptom: Messages like "AuthenticationError," "Incorrect API key provided,"

or similar.
    - Checks:
        1. Double-check that the API key was copied correctly in Step 3A (os.environ["OPENAI_API_KEY"] = "YOUR_API_KEY_HERE").
        2. Ensure that "YOUR_API_KEY_HERE" was completely replaced by the actual key, and the key is within the quotation marks.
        3. Verify that billing details have been added to the OpenAI account and that there are active credits or a valid payment method (Step 2C). API key issues are often related to authentication or billing status.[3]
- **Error like "openai-agents not found" or "Agent is not defined"?**
    - Symptom: ModuleNotFoundError or NameError.
    - Checks:
        1. Ensure the !pip install openai-agents cell (Step 1C) was run successfully.
        2. Confirm that the from agents import Agent, Runner, WebSearchTool cell (Step 3B) was run successfully before trying to define or run the agent. Cells in Colab notebooks generally need to be run in order for their effects (like installations or imports) to be available to subsequent cells.
- **"await outside function" error?**
    - Symptom: SyntaxError: 'await' outside async function.
    - Solution: As mentioned in Step 3D, try combining the agent definition code (the weather_agent = Agent(...) block from Step 3C) and the agent execution code (result = await Runner.run(...) from Step 3D) into the *same* Colab code cell. Then, run this single, combined cell.

D. Explore More! (The AI Journey Continues...)

The newly built weather_agent is ready for more questions! Users can try changing the input to ask about the weather in a different city.

1. In a new Colab code cell, or by modifying the existing one from Step 3D and 3E, try:
```
Python
# To ask about a different city:
# result = await Runner.run(weather_agent, "What's the weather in London?")
# print(result.final_output)
```
Then run the cell(s).

What other weather-related questions could be posed? Remember, its current instructions are "You answer weather questions." Experimenting with different inputs is a great way to understand the agent's capabilities and limitations based on its

configuration.

This tutorial is just the beginning. The AI Unraveled Builder Toolkit will continue with more guides. Look out for the next installment, which might explore topics such as "Giving Your Agent More Tools" or "Making a Story-Writing Agent," to further expand AI building skills.

Works cited

1. An Introduction to Colab and Jupyter for Beginners - Louis Bouchard, accessed on May 28, 2025, https://www.louisbouchard.ai/colab-vs-jupyter/
2. openai/openai-agents-python: A lightweight, powerful framework for multi-agent workflows, accessed on May 28, 2025, https://github.com/openai/openai-agents-python
3. How to Use the OpenAI Agents SDK ? - Apidog, accessed on May 28, 2025, https://apidog.com/blog/how-to-use-openai-agents-sdk/
4. How to Get Access to OpenAI API Keys: A Step-by-Step Guide - Dataaspirant, accessed on May 28, 2025, https://dataaspirant.com/access-openai-api-keys/
5. How to Generate Your Own OpenAI API Key and Add Credits? - Analytics Vidhya, accessed on May 28, 2025, https://www.analyticsvidhya.com/blog/2024/10/openai-api-key-and-add-credits/
6. OpenAI Agents | Phoenix - Arize AI, accessed on May 28, 2025, https://docs.arize.com/phoenix/learn/agents/agent-workflow-patterns/openai-agents
7. New tools for building agents | OpenAI, accessed on May 28, 2025, https://openai.com/index/new-tools-for-building-agents/